CLAIMS

SUB A37

1. A method for designing an electronic system comprising at least one digital part, comprising the steps of :

- representing a behavioral description of said system as a first set of objects with a first set of relations therebetween;
- refining said behavioral description into an implementable description of said system, said implementable description being represented as a second set of objects with a second set of relations therebetween; and
- retaining at least one of said second objects for reuse in the design of a second electronic system.

2. The method as recited in claim 1 wherein said step of retaining comprises the substeps of :

- selecting out of said second set of objects a subset of second objects having substantially the same functionality and/or characteristics in said implementable description;
- creating a class representing said same functionality and/or characteristics; and
- storing said class in a library.

3. The method as recited in claim 2 wherein said second electronic system comprises objects that are instances of said class.

4. The method as recited in claim 2 wherein said second set of objects have a common semantics.

5. The method as recited in claim 2 wherein said class comprises a function.

6. The method as recited in claim 2 wherein said class executes a parametric manipulation on said second set of objects.

7. The method as recited in claim 6 wherein said parametric manipulation is a parametric expansion.

8. The method as in claim 7 wherein said expansion includes the addition of functions to an object for creating a new object.

9. The method as recited in claim 2 wherein said class is a reusable component.

10. The method as recited in claim 9 further comprising the steps of :

- describing the electronic system by formal means in a formal description, said formal description being the representation of said behavioral description of said system as said second set of objects with said second set of relations therebetween;
- selecting a functional entity within said system, said functional entity corresponding to said subset of second objects having substantially the same functionality and/or characteristics in said implementable description ;
- formulating said functional entity as a reusable entity by formulating said functional entity as a parametric expansion of said formal description ;
- describing said reusable entity as said reusable component using said formal description such that said reusable entity is a parametric expansion of said reusable component.

11. The method according to claim 10, wherein said formal description is formulated in an object-oriented programming language, and said parametric expansion is performed on an object hierarchy.

12. The method as recited in claim 2 further comprising the steps of designing another electronic system comprising at least one digital part and wherein said class

is used for creating objects within the design of the other electronic system.

13. The method according to claim 12 further comprising the steps of :

- selecting the behavioral register-transfer level design description of a first hardware component within the design of said electronic system, said hardware component having at least a part of the desired functionality of a target hardware component that is comprised in the design of said other electronic system ;
- determining the changes that are necessary to reuse said hardware component in the design of said other electronic system ; and
- formulating the changes that are necessary to reuse said hardware component in a class that is able to transform the implementable description of said hardware component into said target hardware component.

14. The method as recited in claim 13, wherein said changes comprise a parametric expansion performed on an object hierarchy.

15. The method as recited in claim 14, wherein said object hierarchy is expressed using an object-oriented programming language.

16. The method as recited in claim 15, wherein the object-oriented programming language is C++.

17. The method as recited in claim 13, wherein said behavioral description is described as a hierarchy of one or more objects selected from the group consisting of:

- finite state objects,
- state objects enumerating the states of said finite

state objects,

- transition objects that relate said state objects,
- instruction objects that represent processing done when said transition objects are executed, and
- operation objects that make up parts of said instruction objects.

18. The method as recited in claim 17, wherein the changes are selected from the group consisting of:

- adding extra state objects and/or transition objects to a finite state machine,
- adding extra operations to an instruction objects,
- merging two or more behavioral descriptions,
- removing an object from said hierarchy,
- modifying an object from said hierarchy, and
- any combination of the above.

19. The method as recited in claim 13, wherein the behavioral register-transfer level design of the first hardware component is expressed using an object-oriented programming language.

20. The method according to claim 19, wherein said object-oriented programming language is C++.

21. The method as recited in claim 13, further comprising a refining step, said refining step comprising formulating structural characteristics of a hardware component as an object hierarchy of one or more objects selected from the group consisting of:

- finite state objects,
- state objects enumerating the states of said finite state objects,
- transition objects that relate said state objects,
- instruction objects that represent processing done

when said transition objects are executed, and

- operation objects that make up parts of said instruction objects.

22. Method as recited in claim 21, wherein said refining step comprises the addition of new objects, permitting interaction with existing objects, and adjustments to said existing objects allowing said interaction.

23. Method as recited in claim 21, wherein said refining step is performed in an extendible environment and comprises expansion of existing objects.

24. A method for the reuse of a first hardware component in a hardware design, comprising the following steps:

- selecting the behavioral register-transfer level design description of a first hardware component with at least a part of the desired functionality of a target hardware component that is comprised in said hardware design,
- if necessary, transform said design description to an object hierarchy,
- determine the changes that are necessary to reuse said hardware component in said hardware design, and
- create an object that comprises an expand() method capable of transforming said object hierarchy into a second object hierarchy that describes said target hardware component.

25. The method according to in claim 24, wherein said object hierarchy is expressed using an object-oriented programming language.

26. The method according to in claim 25, wherein the object-oriented programming language is C++.

27. The method according to claim 24, wherein the changes are selected from the group consisting of:

- adding extra states or transitions to a finite state

machine,

- adding extra operations to an instruction to provide extra functionality,
- merging two or more descriptions,
- modifying states, transitions, signals and/or instructions, and
- any combination of the above.

28. The method according to claim 24, wherein the behavioral register-transfer level design of the first hardware component is expressed using an object-oriented programming language.

29. The method according to claim 28, wherein said object-oriented programming language is C++.

30. A method as recited in claim 29 wherein the reuse of a part of a hardware design comprises the steps of:

- describing said hardware design by formal means in a formal description,
- selecting said part of said hardware design,
- formulate said part as a reusable part by formulating said part as a parametric expansion of said formal description,
- describing a reusable prototype of said reusable part using said formal description such that said reusable part is a parametric expansion of said reusable prototype.

31. The method according to claim 30, wherein said formal description is an object-oriented programming language and said parametric expansion is performed on an ~~object~~ hierarchy.